Git is a version control system which is used for software development and management in personal or group projects. The system acts as a central server for developers to manage and add new changes to their project, which can differentiate due to how the group may operate through their workflow. Git consists of functions such as committing code, pushing/pulling code, branching, merging branches, and repository management. Repositories are used for the storage/management of all files used in a personal or group project. Repositories can hold many forms/types of information both locally and remotely, which is very useful for software development. Git allows for developers to create branches of their main project repository to work on their ideas without disrupting the original code. After completing and getting approval of their changes to the branch, developers can also merge their branch into the main repository.

Git commits are essential in software development as it allows for developers to save their progress in the Git history so that developers can record their additions in increments. Commits are helpful when developers create something but end up removing it, the developers can just restore their repository to an older commit. Commits are great to have for situations like those or just for organized, incremental updates which helps keep the repository have a record of progress. Git commits are structured for developers in the order of code editing, change staging, then committing those changes allowing for less error when updating code. Commits allow for developers and future employers to track project development and understand how the development process moved (highlighting features and updates to those features). Git is very effective and commonly attached to group projects and overall project management which is important in organizing the software development process.

GitHub is an online site which provides developers with Git functions so that they can develop, update, and manage their code for personal or group project settings. Git allows for developers to create code to commit to a repository which is then used to keep track of future commits/updates to the code. Developers can write code and retrieve code using the Git functions known commonly as pushing and pulling. Pushing code means to move local repository code changes to another git repository (remote repository) which is how developers update the code they are working on. Pulling code means to extract code/files from the remote git repository to the local repository, altering or adding the files that differ from the remote repository. These two Git commands are essential in software development in group settings because consistent and structured push/pull records allow for controlled development. Controlled and least conflicting development depends on how group members interact with their commits which is essentially their Git workflow.

Git workflow is essential in proper group software development to ensure there aren’t any incorrect changes or merge conflicts occurring within the development process. A commonly used form of Git workflow is the centralized workflow which is commonly used in smaller group projects. Centralized workflow utilizes a central repository as the hub for all changes made to the group project, without the use of branches. Using only the main branch, developers clone the central repository and edit their code as they want, but commits are only stored locally for the project. To further move this code, the developer must push their local main branch to the group central repository which can lead to merge conflicts if not done properly.

Merge conflicts are obstacles in the development process and must be prevented through an organized and well coordinated commit/push pattern. Merging branches occur when the developers don’t update their repositories to new changes and modify the same files changed in a recent update. If the developer doesn’t update their recent group members’ commits, any file lines they modified that the previous commits have modified will cause the merge conflict requiring immediate resolution. The merge conflict can also happen if the previous commit deleted a file which the current developer trying to commit has edited for committing to the main repository. The developers must resolve this merge conflict by deciding what code parts to keep and what parts to remove, using common commands such as Git status, Git add, and Git commit. Git status helps determine the files that are causing the conflict, Git add for staging changes, and Git commit for committing new resolved files. Merge conflicts must be prevented by being more organized and patient with constant software updates, mostly depending on developer discretion. Git, GitHub, and Git workflows are very important in modern software development and should be understood by future developers to elevate their understanding of team development.